Friday
5:30

Kendall Square Research

KSR1 Technology Background

January, 1992

KSR1 Overview

The KSR1 is the first 64-bit highly parallel computer system with the shared memory programming model to satisfy the production requirements of the world's largest users in intensive large-scale numeric processing, on-line transaction processing, and database decision support applications.

ALLCACHE™ is the first memory architecture to deliver the conventional, sequentially consistent shared memory programming model in a highly parallel computer.

The KSR1 is the first highly parallel computer to deliver a mass storage subsystem that is a direct extension of memory (ALLCACHE).

Peak performance ranges from:
- 160 to 21,760 MIPS;
- 320 to 43,520 MFLOPS.

"In each case, we might (cynically?) change the word 'first' to 'first, if any,'. Note that peak MIP rate is 1/2 peak flop rate (on CM-5 integer performance)."
-Guy Steele

"This high end is well below what we can offer. Clearly we have a clear field above ~400 PN's, even if all their claims are true." -Dave Waltz

Balanced scalability in CPU, memory, I/O, mass storage and software is achieved by:

- **Processors that range from 8 to 1088 in number;**
- **ALLCACHE memory that starts at 256 MBytes and exceeds 34,816 MBytes; with virtual memory of 1 Terabyte per process;**
- **I/O bandwidth that ranges from 210 MBytes per second to more than 15,300 MBytes per second;**
- **Maximum disk capacity that spans from 210 GBytes to over 15,300 GBytes.**

The broadest range of traditional software tools on a highly parallel system includes: IBM-compatible COBOL, FORTRAN, ANSI C and fourth generation languages, ORACLE relational database management system, transaction processing monitors, non-relational file access methods, and a fully implemented UNIX operating system (OSF/1 compatible).

Interoperability spans:

- **TCP/IP, NFS, Ethernet) Token Ring, SNA, OSI, X.25, open VME-based interfaces, HIPPI, and FDDI support.**

The KSR1 is a highly available system:

- **RAID storage system ranging from 10 to 15,000 GBytes;**
- **Redundancy array schemes and file striping;**
- **Fault detection and recovery features with redundant and modular components.**

"Ha! They may have COBOL, but we have LISP!" -Guy

[ALLCACHE memory that starts at 256 MBytes and exceeds 34,816 MBytes]
"34,816 MB approximately matches our current 1K machine; any larger CM-5 exceeds this, as will 256 PN and up CM-5's with 16 mbit chips." -DW

[I/O bandwidth that ranges from 210 MBytes per second to more than 15,300 MBytes per second]
"Any CM-5 with 765 addresses devoted to I/O could match this; >765 addresses would exceed this. We could, if we wanted, list much more impressive numbers -- after all, this is just paper and design potential." -DW

[Maximum disk capacity that spans from 210 GBytes to over 15,300 GBytes]
"How much disk could we attach? Pretty much unlimited. Not a very meaningful claim." -DW

[TCP/IP, NFS, Ethernet) Token Ring, SNA, OSI, X.25...]
^^^
"We don't have this -- it would be nice!" -DW

[RAID storage system ranging from 10 to 15,000 GBytes]
"Raid of what level? If RAID-3, ScaleArray will outdo it." -DW

KSR1 Computer System

"The KSR1 is the first highly parallel computer system that unites the power of parallel processing with a standard shared memory software development environment, to satisfy the production requirements of commercial and technical applications. The first family of Kendall Square Research products, the KSR1 highly parallel computer is composed of large numbers of RISC-style superscalar 64-bit processors. The system scales from 8 to 1088 processors. The KSR1 offers a performance range from 160 MIPS (millions of instructions per second) and 320 MFLOPS (millions of floating point operations per second), to 21,760 MIPS and 43,520 MFLOPS."

"Kendall Square Research has achieved three major industry breakthroughs in hardware and software design: the ALLCACHE memory management architecture that brings the traditional shared memory programming model to parallel programming, raising its efficiency and simplicity to a higher level; the combination of conventional software tools and a highly scalable parallel machine that empowers users to address a broad range of applications in commercial and technical environments; and a price/performance curve that introduces "desktop economics" to the traditional "Glass House' of central servers."

[traditional shared memory programming model]
^^^^^^^^ *"???" -Guy*

[...to satisfy the production requirements of commercial and technical applications]
"A non-sequitur. Commercial and technical applications can be satisfied other ways." -DW

[...brings the traditional shared memory programming model to parallel programming, raising its efficiency...]
"Why? Not at all obvious. Requires testing/performance evaluation." -DW

[...and simplicity...]
"Perhaps." -DW

['Glass House']
"What does this mean?" -DW

"With these breakthroughs, Kendall Square Research is targeting the production requirements of the world's largest users in three key areas: large-scale numerically intensive processing, on-line transaction processing (OLTP), and database decision support applications, in any combination. As a productivity tool, the KSR1 enables hybrid processing that involves two or more of these modes simultaneously, especially important in complex applications such as on-line trading systems, and multi-site materials requirements planning (MRP) systems."

[large-scale numerically intensive processing, on-line transaction processing (OLTP), and database decision support applications,.in any combination.]

"They haven't claimed, let along demonstrated, operating systems advances. Do we really believe these can be combined easily/smoothly? No, not today. Nonetheless, a powerful combination if they can pull it off." -DW

Historical Trade Offs

"Over the past two decades, large-scale users have increasingly required greater computational power and higher levels of performance for numerically intensive, OLTP and database applications. In response, computer designers have primarily pursued two general approaches. Certain companies, including Cray Research and IBM, have designed high performance computers by implementing one or a small number of the fastest single processors available. Improvements in performance were achieved by designing processors with components having faster switching times and greater densities and, in the case of numerically intensive applications, by employing vector processing. Vector processing applies a small set of program instructions repeatedly to multiple data elements. Vector processing has proven to be highly effective for certain numerically intensive applications, but not effective for OLTP or database applications. Moreover, these computer systems are subject to the inherent physical limitations of single-processor performance (such as the speed of light and the laws of thermodynamics), and to architectural constraints on the number of processors within a system."

"Paul R. has pointed out that OLTP requirements are probably met for speed today, though size and complexity of transactions may be limited." -DW

[Vector processing has proven to be highly effective for certain numerically intensive applications, but not effective for OLTP...]
"Agree" -DW

[...or database applications.]
"Probably not true. While Cray hasn't done this, vector processing probably could be used effectively here, at least for a range of tasks." -DW

"Other computer companies, including Thinking Machines and Intel, have focused on parallel processing as a way to achieve greater computational power and to improve price/performance characteristics. Parallel processing enables large numbers of processors to act concurrently on multiple tasks or in concert on single computationally- intensive tasks. The systems use large numbers of low cost processors, taking advantage of dramatic improvements in semiconductor price/performance characteristics. However, these computer firms have experienced limited market acceptance because their distributed memory architectures cannot support conventional shared memory programming. Virtually all existing algorithms, computational science, OLTP and database applications, and supporting software were developed for shared memory architectures. In the shared memory architecture of conventional supercomputers and mainframes, processors are coupled to a single shared memory, which facilitates the sharing and synchronization of data without requiring the development of special memory management software."

"Again, shared memory programming is anything but conventional. We should hammer on this. Has it really been successful? Has KSR solved the problems when scaled to 1088 processors?" -Guy

[Virtually all existing algorithms ... were developed for shared memory architectures.]
"with small N. Will this technology carry over as smoothly as KSR claims? For example, a simple task-queue algorithm might work fine on 8 processors, but maybe on 800 processors the queue becomes a bottleneck if you naively run the same code."

-Guy

"Through 1985, this was largely, though not universally true. Since then, there has been <u>lots</u> of effort and progress on distributed memory parallel algorithms."

-DW

[the sharing and synchronization of data]
"What is synchronization of data?" -DW

"Indeed, programming problems stemming from today's highly parallel memory architectures are reminiscent of storage management difficulties in the 1960's. Then, storage management was an integral part of writing a program. The responsibility fell upon the developer to perform a static analysis of a program's memory requirements, to split the program into "overlays," and to swap the overlays explicitly in and out of main memory. Advances in computer engineering, however, gradually rendered these methods impracticable. The introduction of multi- programming systems, for instance, made it infeasible for the developer of a single program to prediet accurately the time-varying storage requirements of an entire system."

"I can't say whether this is true or not, fair or not." -DW

"Ultimately, these issues led to the adoption of virtual memory as a near-universal feature of storage management in modern computer architectures. Virtual memory made storage management dynamic and largely automatic. It permitted programmers to write applications with a storage abstraction which was simple and powerful -- a single uniform address space.

"Though not universal in supercomputers! Seymour Cray has explicitly said paging is a bad idea in his realm." -Guy

"Misleading. This certainly ignores the fact that DB and OLTP applications do not use virtual memory -- they may use inverted files, but they don't have the whole DB as a shared address space! In fact, I don't think virtual memory has been used widely for Cray-type applications either. All these tasks -- the ones they say they're addressing, do not use virtual memory, I believe (get a second opinion though)."
-DW

"To achieve high performance, developers of highly parallel machines have used distributed memory architectures across a multi-computer system. That is, physical memory is composed of a set of memory units, each connected to a unique processor. The processor-memory pairs are interconnected by a network."

"This has also been -- in the case of Intel, NCube, and CM-5 -- to allow the use of standard chips, that get their speed from caches, without having to solve difficult cache-coherence problems. This (not lack of virtual memory) is what limits size of bus machines (Convex, Sequent, etc.)." -DW

"However, as in the 1960's, the job of managing the movement of programs and data among these distributed memory units has generally fallen upon the programmer. Developers must worry about what will fit where and what to remove to make room for something new. The task is similar to managing the migration of data back and forth between primary and secondary storage devices prior to the introduction of virtual memory. But memory design is now highly complex; the developer must deal with thousands of memory modules, not just two or three."

"But TMC solves this problem with compilers rather than burdening the user."
-Guy

"We've certainly made lots of headway. However, OLTP is still not easily doable. With ScaleArray and software, it will be possible." -DW

[But memory design is now highly complex; the developer must deal with thousands of memory modules, not just two or three.]
"This falsely suggests that the programmer deals with them individually, which is completely untrue for data parallel programs." -Ephraim Vishniac

"Because of the high cost of developing software for most highly parallel computers, these systems have been used for only one or a few applications. Although some large-scale users have developed technical and engineering software for certain parallel applications, users have not found it practicable to port a large number of third-party software applications on to these systems. The absence of a conventional software development environment, or widely available third-party applications programs, has severely restricted acceptance of parallel processing computers."

[absence of a conventional software development environment]
"baloney" -Guy

[these systems have been used for only one or a few applications]
"Clearly a self-serving and gross exaggeration!" -DW

"It may be that third parties hadn't seen enough copies of the machines being made to justify the ports (or perhaps we've been too compacent or ineffective in signing up third parties)." -DW

[...has severely restricted acceptance of parallel processing computers.]
"Until recently we haven't even tried to hit more than a few selected applications."
-DW

[the high cost of developing software for most highly parallel computers]
"Actually, the cost of developing <u>any</u> software is high. That's why folks are wedded to their dusty decks." -Ephraim

"I think this may be true. We should improve our software environment to make it easier for the user (or third party) to port other software.

"The very large database is a good application field. I think we should decide what we should do. Right now we are looking for third party to work together on this project. How can we speed up this process? Can we have some kind of research group to concentrate on developing our own parallel database processing algorithm, and in the meantime have some group search for third party and interface with other standard database systems? I guess in this parallel way the CMDB (CM Database) would be speeded up. Our group should have a product, otherwise we are in a passive position. We have to break up to help other busy groups." -Lily Li

"Users of high-performance systems, therefore, face a limited choice. Supercomputers, mainframes, and vector processing systems, permit a conventional programming environment, but are intrinsically limited in performance and are burdened with a higher cost of computation; parallel processing systems offer very high performance and lower computation costs, yet are difficult or impractical to program.

"We can certainly dispute this, at least to a degree." -DW

"Kendall Square Research is addressing this dilemma by developing a highly parallel computer system that combines the high performance and lower computational costs, inherent in parallel processing, with the simplicity of conventional shared memory programming and the benefits of emerging industry standards. To implement this strategy, the company has created and patented a major technological innovation in computer
architecture-- the ALLCACHE memory system."

"We have standards on our side, too -- Fortran 90, C(?), Unix, HIPPI, etc." -DW*

ALLCACHE Memory

"ALLCACHE memory creates a familiar and easy-to-use development environment that eradicates the plethora of programming difficulties associated with highly parallel processing systems. ALLCACHE automates the addressing and location of memory. Users can port existing application programs to the KSR1, and develop new applications using industry standard development environments."

"The key question is: Can they really pull this off at a competitive price 1) today, and 2) as the industry develops? Ksr has introduced a heavy hardware overhead on their system, one that won't get cheaper automatically by competition (as will SPARC chips, 486's, etc.). It will be interesting to see how expensive their system will really be. There is no question that software engineering is terribly expensive. However, once applications are ported to distributed memory, the future looks bright for us. If KSR's hardware is cheap enough, they will get a strong short-term boost. They may have trouble competing in the future though." -DW

"ALLCACHE is the first memory architecture to deliver the conventional, sequentially consistent shared memory programming model in a highly parallel computer. ALLCACHE combines this memory model, used by traditional supercomputers and mainframes, with the scalability of highly parallel systems. Scalability allows users to add computer resources in incremental and cost-effective steps, without changes in software and without performance degradation. Systems that implement sequential consistency guarantee that a program behaves in the most intuitive manner to the programmer: the result of a program execution on parallel processors is equivalent to the execution of the program on a multi-tasking single processor, which carries out tasks in a sequential fashion."

"As far as I know, writing software wich takes advantage of multi-tasking capabilities of existing systems (VAX or IBM-mainframe?) is extremely difficult -- more difficult than writing software for CM-2(5). -Alex Kushkuley

[sequential consistency]
"Here's one to focus on. They are glibly thowing around this technical term to make it appear that you get the same results on a completely sequential (e.g., Fortran 77) program. But this refers to the fact that each memory location behaves as though it had received concurrent read and write requests in some sequential order -- but there can still be race conditions relative to multiple memory loctions. So it is not as simple as programming a sequential machine." -Guy

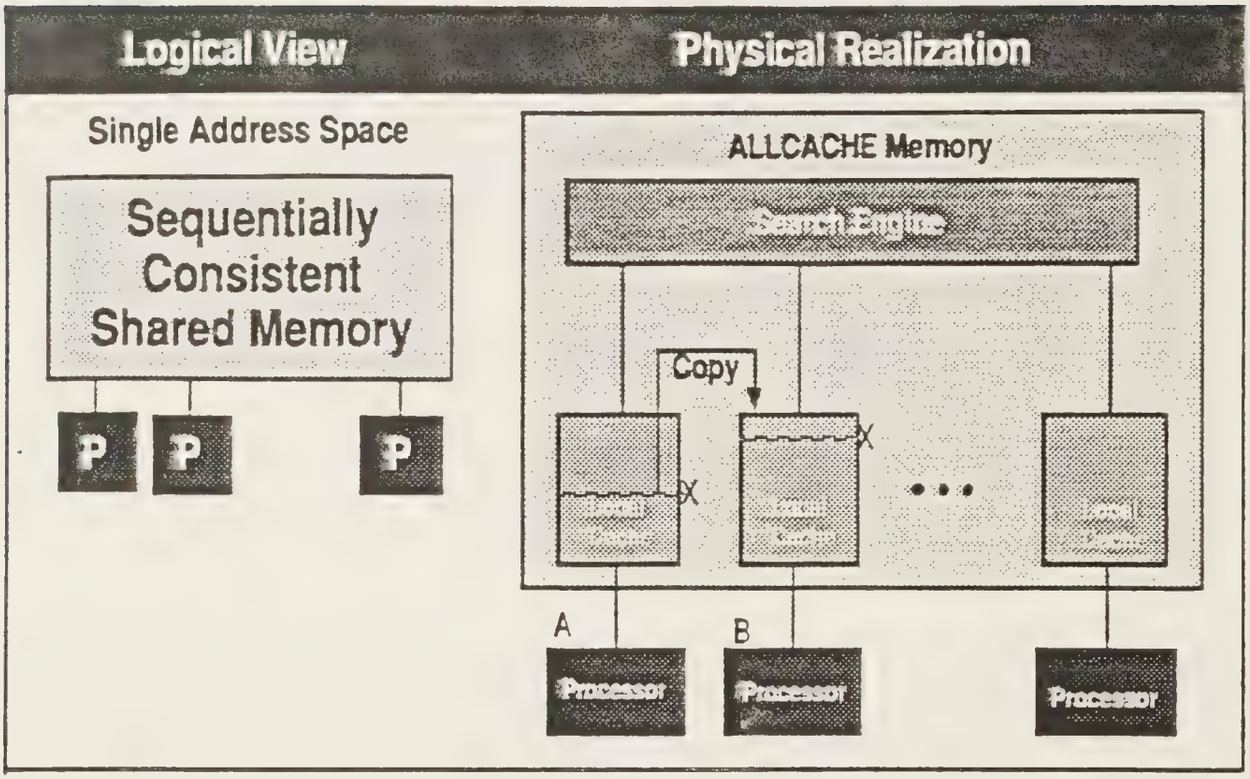
[carries out tasks in a sequential fashion]
"interleaved" -Guy

[with the scalability of highly parallel systems]
"It still remains to be seen how scalable this scheme is. Network latency and throughput seem likely to be problems." -DW

[without performance degradation]
"If the best they can say is that more processors = not slower, then we have nothing to worry about! But I doubt they mean this." -DW

[a program behaves in the most intuitive manner to the programmer]
"If he's been von Neumann-brainwashed. As you (Jim B) have shown, data parallel is (also) intuitive." -DW

Figure 1 ALLCACHE Memory System



"ALLCACHE is - literally - all cache. Thirty-two MBytes of cache memory are attached to each KSR1 processor. That's the only memory there is. No permanent physical location exists for an "address," as illustrated in Figure 1. Rather, addresses are distributed and shared based on processor need and access patterns. To the application developer, memory is a single resource. While Kendall Square Research has designed the KSR1 architecture to support a 64-bit address space (2^{64}), the current KSR1 implementation supports a 40-bit address space (2^{40}) -- providing one Terabyte (one trillion bytes) of virtual address space per process. This greatly exceeds the memory capacity required in today's high performance computing environments."

"Is this true of any size machine -- 8 to 1,000 processors?" -DW

[virtual address space per process]

"How many processes are possible?" -DW

"ALLCACHE leverages a property of address reference sequences called "locality of reference." Instructions and data, once referenced, are likely to be referenced again. When addresses are requested by a processor through the execution of a load, store or branch instruction, ALLCACHE automatically moves them to the processor's physical "neighborhood" (a 32 Mbyte cache memory module). ALLCACHE keeps memory traffic close to the area that is using it. This reduces latency (travel time) and avoids communications congestion."

[ALLCACHE is - literally - all cache]
"How fast is this memory? If it's the usual 60ns or so DRAM, then this is merely a clever terminology gambit - merely slapping the name 'cache' on a fairly ordinary main memory doesn't make it a cache. On the other hand, if it is special fast memory, then it is too expensive.

"I say ALLCACHE is just an ordinary main memory with a clever MMU and network. Question: is putting the cleverness n hardware rather than software worth the cost of silicon and inflexibility?" -Guy

"So -- suppose two processors are using the same datum. Does it get moved back and forth between the two processors as they each reference it? If not, some process has to take care of coherence." -DW

[This reduces latency (travel time) and avoids communications congestion.]
"Only if the case above doesn't occur." -DW

"This picture is too optimistic. Many problems cannot be solved by 'local' computations. There are well known cases (British Museum-type problems) when the best search strategy is to 'visit' each data element exactly one time (i.e., any 'optimization' will make the search slower). 'Random' transaction processing also cannot be optimized in the suggested way. On the other hand, for the algorithms which do exhibit such local properties, the best hardware architecture is probably that of CM-2." - Alex Kushkuley

"The KSR1 memory architecture consists of two levels of address space: Context Address (CA) space and System Virtual Address (SVA) space. CA space is the programmer's interface to memory. There are many CA spaces in a system. SVA space stores the data from all context address spaces. There is only one SVA space. Load, store and branch instructions generate CAs, which are translated by the processor into SVAs.

"To coordinate this high level of memory interaction, the KSR1 employs a Search Engine. Attached to each KSR1 processor, the Search Engine is a distributed mechanism responsible for efficiently finding addresses and their contents, relocating them to a processor's local cache when referenced by that local processor, and maintaining sequential consistency between caches. The address and data remain at the local cache until the space is needed by another address."

"The Search Engine interconnects local caches and provides routing and directory services for the collection of local eaches, while maintaining coherence throughout the system. As a result, the collection of local caches behave as a single shared address space.

[Search Engine]
"Gee - doesn't that sound slow? :-)" -Guy

"Ask about latency." -Guy

[The Search Engine interconnects local caches and provides routing...]
"Sounds like a network to me.
One might compare this architecture to a BBN Buterfly (not identical, but similar)." -Guy

"Again, only a good idea if it's not expensive, and if there aren't screw cases (e.g., global variables?)." -DW

"However, providing a shared memory architecture on a parallel computing platform is not adequate by itself. The challenge is to build a shared memory model that allows the system to scale to large sizes. ALLCACHE's scalable interconnection model is a two-tier hierarchy of "search groups" composed of clusters of processors and memory."

"The first level, called Search Group:0, consists of clusters of up to 32 processors. The second level, called Search Group:1, consists of up to 34 Search Group:Os, or 1088 processors. In the future, this clustering technique can be applied at additional levels of hierarchy to build systems with tens of thousands of processors. With two levels, KSR1 performance greatly exceeds that of any mainframe or supercomputer in the market today.

"It's easy to claim, but may be harder to do." -DW

[With two levels, KSR1 performance greatly exceeds that of any mainframe or supercomputer in the market today.]

"CM-5 is a counter example - unless we are quibbling over the word 'supercomputer.'" -Guy

"Are we a supercomputer? If not, I suppose this could be true, but it's certainly misleading!" -DW

"The ALLCACHE design eliminates the traditional parallel bottlenecks in memory traffic, provides a simple programming model, and supports an entire environment of familiar software tools. Users can preserve traditional programming approaches, while utilizing the price/performance advantage of highly parallel systems to build the next generation of high performance computing solutions."

[traditional parallel bottlenecks]

"Strange turn of phrase." -Guy

"What are these? They must be referring to BBN, Sequent, Convex, etc. Not us." -DW

"Oxymoron?" -Ephraim

[price/performance advantage of highly parallel systems]

"Again, only if ALLCACHE doesn't add prohibitively to cost, now and/or later." -DW

"Having read this, I'm still completely uninformed about how the system efficiently handles variables accessed by many processors. In a system that's actually serial, there's no problem -- the variable tends to be cached. In a parallel system, won't such globals effectively serialize operation? Then you need a data parallel re-write, and then you might as well buy a CM-5..." -Ephraim

The APRD Cell

"At the heart of the ALLCACHE memory architecture is the KSR1 ALLCACHE Processor, Routing and Directory cell (APRD cell), which includes a 64-bit superscalar processor, 32 MBytes of cache memory, and the Search Engine. The Search Engine is implemented in the APRD cell's hardware.

"The APRD cell is composed of six types of full custom complementary metal-oxide semiconductor (CMOS) chips, packaged on a 20-by-32 centimeter (8-by-13-inch) circuit board, as noted in Figure 2. Using 1.2 micron CMOS, each KSR1 custom chip contains up to 500,000 transistors. The processor is clocked at 20 MHz and executes 2 instructions per cycle."

"The processor portion of the APRD cell is implemented with four chips: the Cell Execution Unit, generating addresses; the Floating Point Unit, performing arithmetic operations on 64-bit IEEE floating point numbers; the Integer Processing Unit, executing arithmetic and logical operations on 64-bit integers; and an External I/O Unit, moving data between peripheral devices and ALLCACHE memory. Combined, the processor delivers a performance that ranges from 6.6 MFLOPS (Livermore loop harmonic mean), to 14.5 MFLOPS (100 X 100 Linpack), to 28 MFLOPS (FFT), to 32 MFLOPS (Matrix Multiply)."

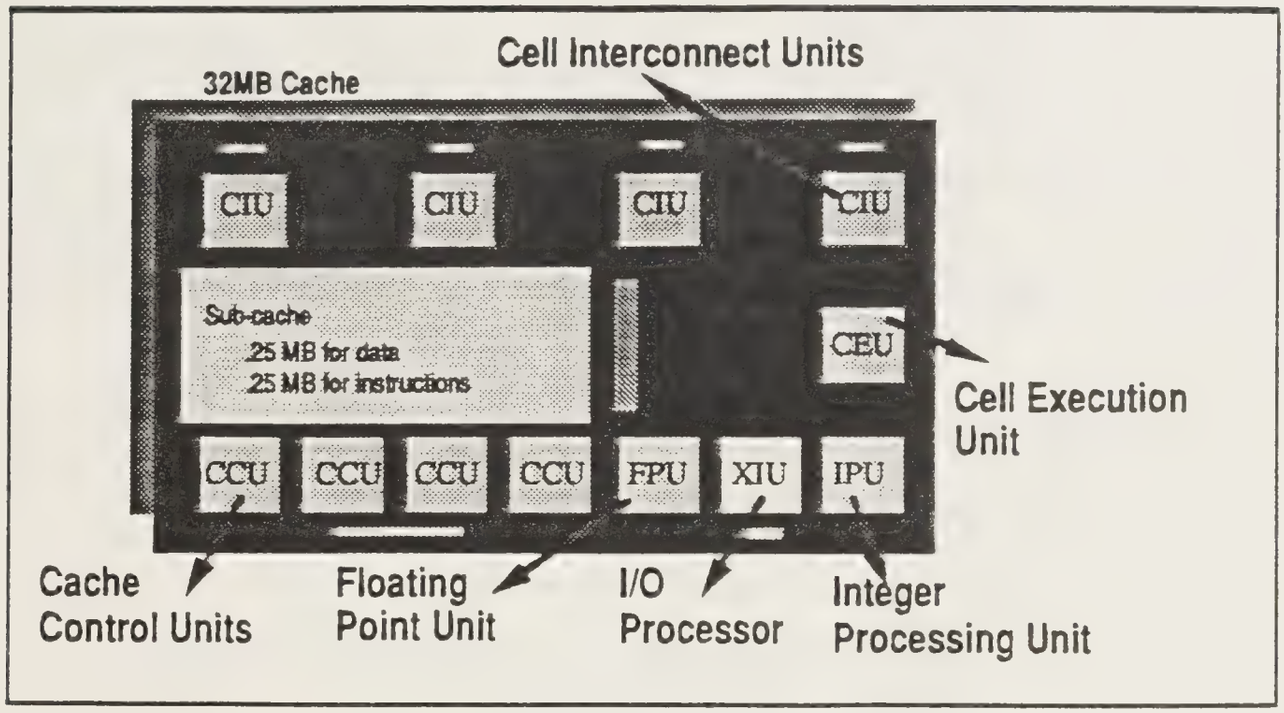
[The processor is clocked at 20 MHz and executes 2 instructions per cycle.]

"Probe this. Does this mean 1 instruction/cycle, but floating mult-add is one instruction? Or what?" -Guy

[100 X 100 Linpack]
"about 3 X a plain SPARC (as used in CM-5)" -Guy

[the Cell Execution Unit, generating addresses]
"?" -DW

Figure 2 The APRD Cell



[Sub-cache/ .25 MB for data/ .25 MB for instructions]
"Aha! I was right! They rename "main memory" as "cache" and call the real cache a "subcache"! 256 Kbytes each for I and D (here labelled .25 MB!)

"What is their network topology? What is bisection bandwidth?" -Guy

"Underlying this performance is a 40-bit address space, IEEE- standard 64-bit floating point format, a peak floating point rate of 40 MFLOPS, and two 64-bit buses for the transport of data and instructions.

"So what's the programming model? Where is the (a) program stored?" -DW

"In addition to the 32 MBytes of dynamic random-access cache memory, each APRD cell has two 256 KBytes of static random access sub- cache memory. The APRD cell also has large register sets to reduce memory traffic: sixty four 64-bit floating point registers; thirty two 64-bit integer registers; and thirty two 40-bit address registers."

"Decent." -Guy

"Address translation begins when a memory or branch instruction calls for a Context Address (CA). The processor uses an address register a signed displacement to produce 40-bit CAs, which are then translated into 40-bit System Virtual Addresses (SVA). The mapping from a CA space to the SVA space is implemented using Segment Translation Tables which are managed by system software. The process is transparent to application programs.

"This doesn't really explain how the system works well enough to understand it." -DW

[Segment Translation Tables]
"How big are these tables? Are they cached also? How often does it fault?" -Guy

"Inter-cell traffic travels on pathways with capacities of 1 to 4 Gigabytes per second. Each APRD cell's external I/O channel operates at a sustained rate of 30 MBytes per second."

"So how fast can data be transferred onto the pathway? That may be more relevant." -DW

"Their maximum memory bandwidth per node is 160Mbytes/sec, assuming that the data is in local cache. This is 1/3 our memory bandwidth with the VUs.

"They have 40-bit virtual addresses in the first implementation. They also talk alot about mapping files into virtual address space. With the 40-bit addresses the largest files they can have are 1Tbyte with this technique." -Dave Douglas

"I am not sure, but 30MB/sec is comparable to CM-5 network and to optical LANs -- if the underlying network hardware can be made faster -- then CM-5-type architecture will utilize the advantage. I am trying to say that it is better to have 'network architecture' to which the processors can be 'plugged in.'" -Alex Kushkuley

Design Strategy

"ALLCACHE's architectural solution required the development of custom CMOS chips. Standard "off-the-shelf" chips available from semiconductor manufacturers are designed for desktop solutions. They could not support ALLCACHE's functional requirements, such as 40 bits of address space. Additionally, these chips could not achieve KSR1's price/performance expectations, nor support a processor scheme that is highly integrated with memory. For example, the KSR1 processor allows two or more processors to synchronize their access to the same block of memory through the use of sophisticated locking and unlocking functions."

"That is programming style which makes debugging of complex programs very difficult. Programming with 'semaphores' is a paradigm which probably was invented before any of the massively parallel computers existed and doesn't take into account new possibilities." -Alex K

[custom CMOS chips]
"Therefore not compatible with existing software as SPARCs are" -Guy

[these chips could not achieve KSR1's price/performance expectations]
"This seems preposterous! No small volume custom chip (with special extra features) is likely to be able to compete on price/performance with commodity workstation/PC chips." -DW

[For example, the KSR1 processor allows two or more processors to synchronize their access to the same block of memory through the use of sophisticated locking and unlocking functions.]
"[This] is really bullshit. Every microprocessor designed in the last 3 years has this capability. They made statements like this a couple of times." -Dave Douglas

"Kendall Square Research's strategy has been to invest in the development of proprietary, high performance processors. These processors support the KSR1 system's advanced architecture, while drawing on commercially available technologies to reduce costs, facilitate high volume production and increase reliability. The company believes that its investment in the CMOS-based semiconductor devices, which make up the APRD cell, is an important competitive advantage which others cannot duplicate quickly or easily."

[increase reliability]
"First mention. What has been done about reliability? It doesn't seem to have played a very prominent role in the design." -DW

"We'll see about price performance." -DD

"Kendall Square Research is the first company to introduce a 64-bit floating point processor cell for high performance applications based on CMOS technology. The firm's ability to develop and economically fabricate semiconductor devices has enabled Kendall Square Research to implement the ALLCACHE memory system architecture, and to achieve very high processor density, reducing board size and production costs. At the same time, the use of CMOS technology permits the company to continue to leverage the dramatic increases in the economics of commercial CMOS production. These advantages are especially evident when compared to expensive process technology, such as ECL or Galium Arsenide, often employed in building high performance computer systems."

[Kendall Square Research is the first company to introduce a 64-bit floating point processor cell for high performance applications based on CMOS technology.]
"Hah!!!!" -DD

"Kendall Square Research also benefits from the reduced heat generation and high reliability associated with CMOS technology. KSR1 chips are fabricated by the Japanese consumer electronics giant, Sharp Corporation, whose chips drive thousands of consumer product types such as copying machines, video recorders and hand-held computers. The KSR1 APRD cells are packaged using low cost and reliable tape-automated bonding (TAB) packaging. Kendall Square Research assembles all of the APRD Cells. Assembled boards, modules and adaptors are then individually tested by the company. Final assembly and testing of each system configuration is based on specific customer orders."

[KSR1 chips are fabricated by the Japanese consumer]
"All CM-5 chips are multiple-sourced; each part has at least one American source. (I heard Danny say this once -- still true?)" -Guy

"Note that all of their chips are fabbed in Japan, we should use this against them in the US." -DD

[high reliability associated with CMOS technology]
"Is this the entire basis for reliability claims? We can surely show much more effort/support for reliability." -DW

[KSR1 chips are fabricated by the Japanese consumer electronics giant, Sharp Corporation, whose chips drive thousands of consumer product types...]
"Hype" -DW

"The result is a system architecture capable of achieving very high performance and scalability characteristics, while drawing upon technologies which are proven, produced in large volumes, and do not depend upon special cooling, large spaces, or support. Installation of KSR1 systems is routinely completed in under four hours.

[Installation of KSR1 systems is routinely completed in under four hours.]
"Bsaed on a very small sample of very small machines!" -DW

"The KSR1 computer system, highlighted in Table 1, will drive new price/performance and scalability standards in high performance computing. For example, a 16-processor KSR1 system, called KSR1-16, provides similar processing power to that of an IBM ES9000-620 mainframe, at 1/8 the price, 1/10 the size, and 1/30 the power consumption, driving down the cost of ownership significantly."

"Service, system administration, support, training, etc. will all have to be good; or this has little chance against IBM, or us. Not really a big enough price/performance advantage to insure success. I believe we have closer to a 100:1 advantage over mainframes, and much more developed support, plus a track record." -DW

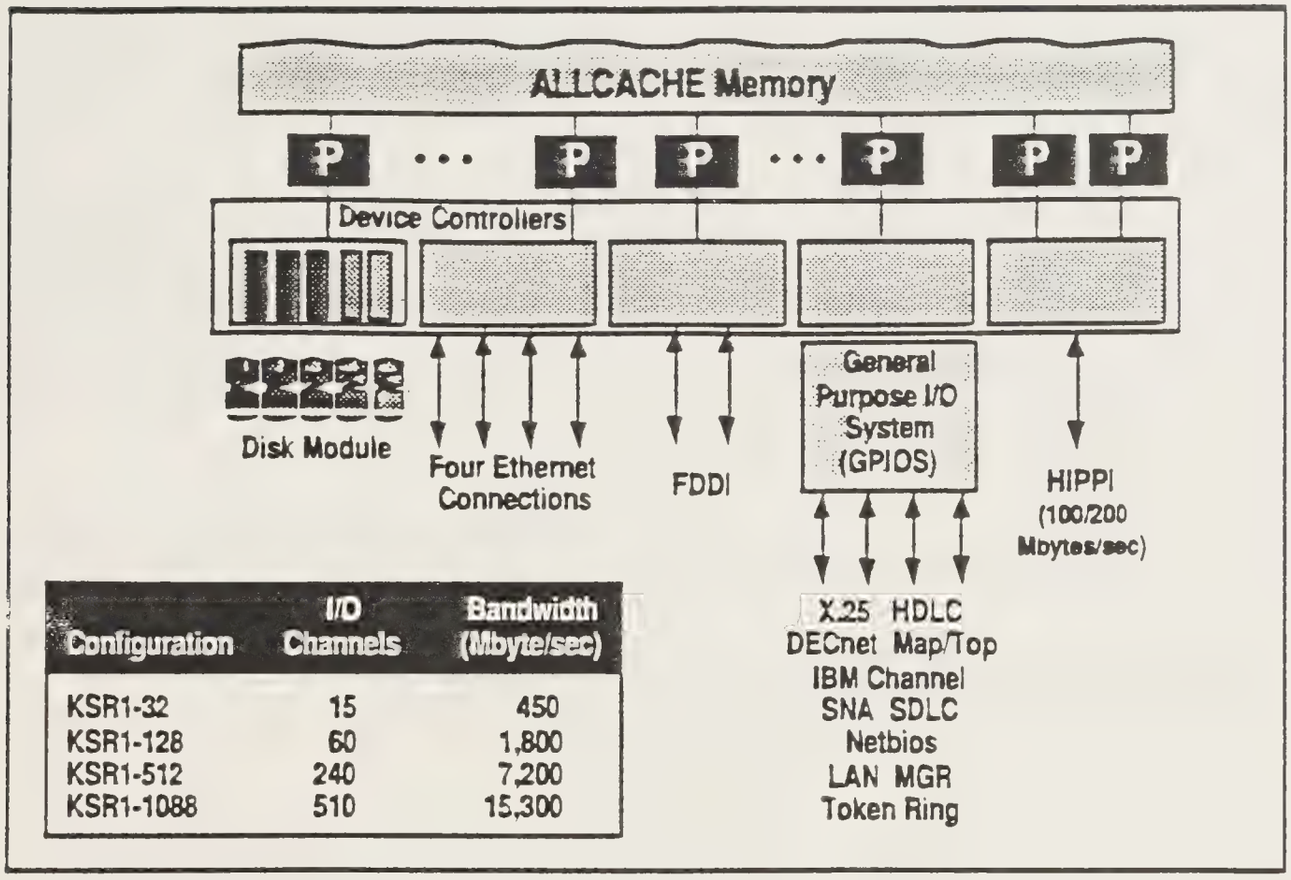
"The impact of the KSR1 design is additionally evident in its capacity to offer balanced scalability that ensures seamless growth without rewriting code or impacting the system's performance. The KSR1 architecture supports scalability in all critical dimensions of the computing process, including compute power, memory, I/O bandwidth, mass storage and software. Figure 3 illustrates scalable I/O bandwidth."

Table 1 Performance, Scalability and Balance

Processor Config-urations	MIPS	Peak MFLOPs	Memory (MBytes)	Max. Disk Capacity (GBytes)	Max. I/O Capacity (MBytes/sec)
KSR1-8	160	320	256	210	210
KSR1-16	320	640	512	450	450
KSR1-32	640	1,280	1,024	450	450
KSR1-64	1,280	2,560	2,048	900	900
KSR1-128	2,560	5,120	4,096	1,800	1,800
KSR1-256	5,120	10,240	8,192	3,600	3,600
KSR1-512	10,240	20,480	16,384	7,200	7,200
KSR1-1088	21,760	43,520	34,816	15,300	15,300

"We have a clear advantage in not being limited in disk capacity, even at small machine sizes. However, max I/O with ScaleArray will be similar on machines with similar numbers of PN's." -DW

Figure 3 Scalable I/O Bandwidth



Configuration	I/O Channels	Bandwidth (Mbyte/sec)
KSR1-32	15	450
KSR1-128	60	1,800
KSR1-512	240	7,200
KSR1-1088	510	15,300

"Note: I/O capacity not independent of processor capacity?

"Doing I/O to remote processors may disrupt operations of processors to which I/O devices are attached?" -Guy

KSR1 General Purpose Programming Environment

"On this powerful platform, Kendall Square Research has introduced traditional software tools that allow the KSR1 to be programmed in an open environment that follows industry standards.

"The KSR1 is the first highly parallel computer to run a full implementation of the UNIX operating system. KSR OS™, the KSR1 operating system, is an enhanced version of the Open Software Foundation's OSF/1. KSR OS lets programmers develop, port and run applications easily, as well as dynamically allocate system resources through standard UNIX capabilities. This includes the ability to timeshare processors for better utilization -- especially important in interactive jobs such as scientific visualization and commercial transaction processing."

"If all is based on UNIX, I'd expect problems in fielding production OLTP -- UNIX has poor facilities for data integrity, security, etc." -DW

"Note that they are running a full Unix on every node, not the microkernel for distributed machines that OSF is doing for Intel." -DD

[full implementation ... an enhanced version]
"These don't seem to match." -DW

[The KSR1 is the first highly parallel computer to run a full implementation of the UNIX operating system.]
"baloney" -Guy

"KSR OS is compatible with AT&T's System V, Release 3, Berkeley Systems Distribution 4.3, X/Open XPG3, POSIX 1003, and FIPS 151.1. The KSR1 system also supports window-oriented user interfaces built around X-Windows and Motif.

"The KSR1 supports standard programming languages including IBM-compatible COBOL, Fortran, with automatic parallelization, and ANSI C. The ORACLE relational database management system (RDBMS), including application development tools, will be available on the KSR1 in 1992. Kendall Square Research is extending ORACLE's features for the parallel environment. Figure 4 depicts COBOL and ORACLE for KSR1 within the broader commercial environment. Additionally, the KSR1 will support the TUXEDO transaction processing (TP) monitor, fast non-relational file access methods, and fourth generation languages."

[Kendall Square Research is extending ORACLE's features for the parallel environment]
"If code runs unchanged on DSR, why was it 'extended'? (This is, I suspect, a really important questions.)" -DW

"For decision support applications, ORACLE for KSR1 will provide automatic parallel processing for complex queries. Kendall Square Research has developed a general purpose technique called Query Decomposition which automatically parallelizes SQL queries generated by ORACLE-based applications. Future third party RDBMS software ported to the KSR1 will also take advantage of the Query Decomposition tool."

[technique called Query Decomposition which automatically parallelizes SQL queries generated by ORACLE-based applications]
"Same question (as above)" -DW

"The notion that 'Shared Address Space' was helpful here is bunk. The notion that they are somehow running software written for a shared memory machine is bunk.

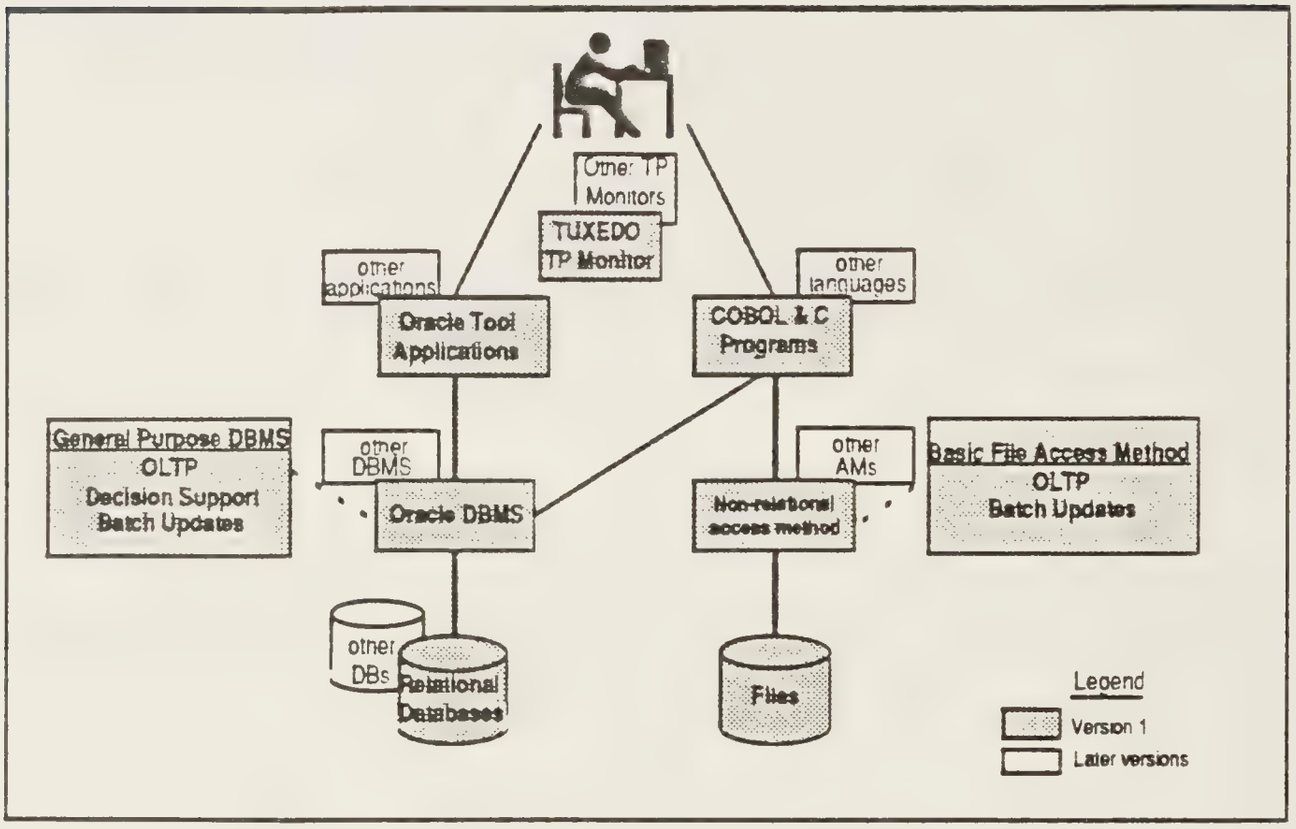
1) The Oracle software was written for the N-Cube, a distributed memory machine. Shared Address Space is not needed for this software.

2) The 'automatic parallel decomposition' is a further illustration that there is no free ride to parallelism. In fact, the Oracle software, as provided, has no capability for parallelizing queries. KSR had to add it by writing new code.

And so, their most highly touted 'success stories' will be 'successful' (when and if they work) for reasons that have absolutely nothing to do with ALLCACHE."

- Craig Stanfill

Figure 4 KSR1 Commercial Software Environment



"While offering a highly parallel applications development environment, Kendall Square Research will make available in 1992 scientific and mathematical subroutine libraries, and software packages for computational fluid dynamics, quantum chemistry, mathematical algorithms for engineering applications, molecular dynamic modeling for computational chemistry, and finite element analysis for engineering applications.

"In addition to supporting a wide range of technical applications, the KSR1 can also act as a centralized, high performance graphics server for low-cost workstations. Because of KSR1 performance, users can explore "motion pictures" of application models in scientific and commercial graphically-oriented programs."

"KSR1 co-exists in multi-vendor environments, diagrammed in Figure 6. KSR1 supports an extensive set of connectivity technology including:

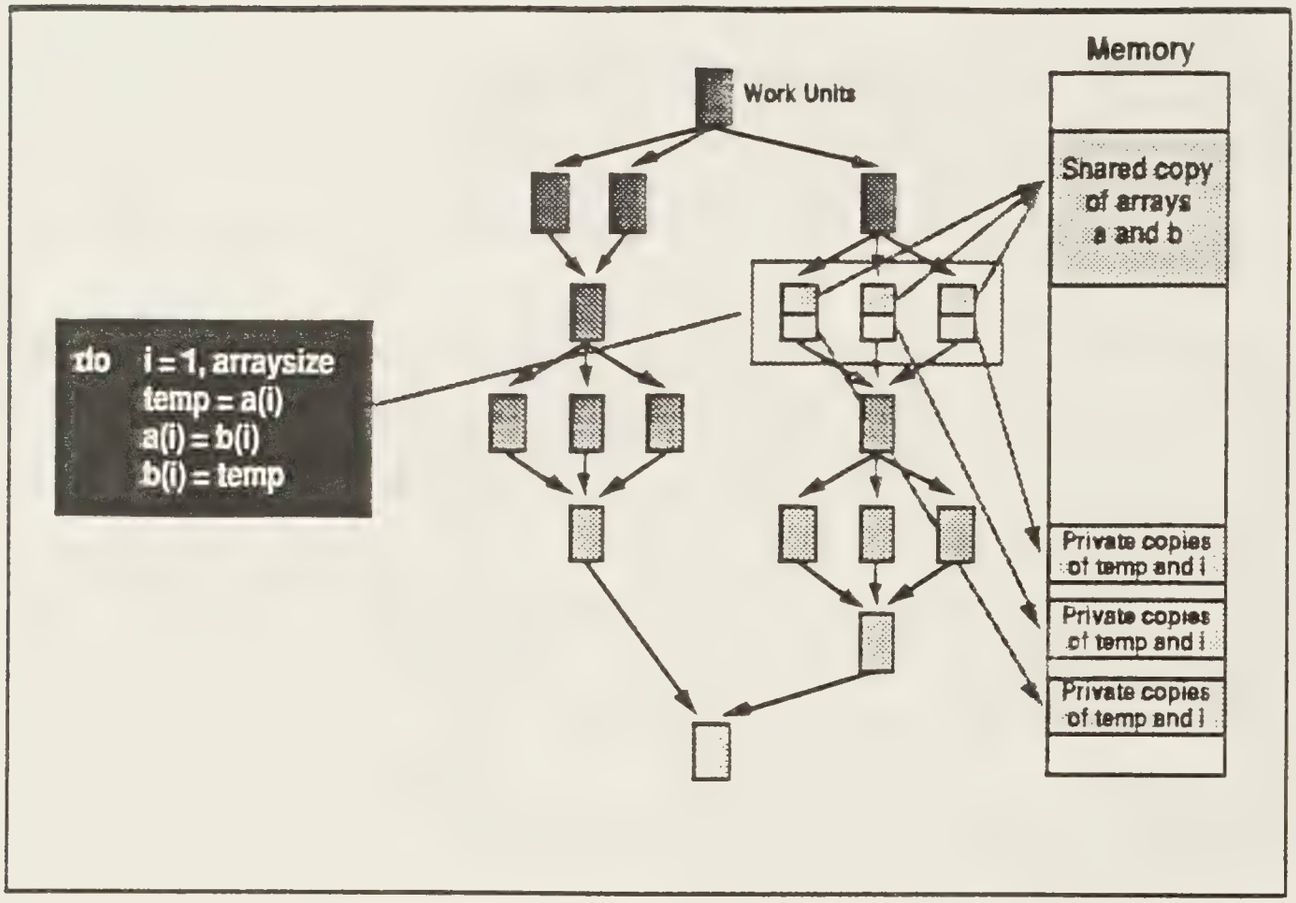
• TCP/IP, NFS, SNA-3270, 3770, LU6.2/PU2.1, International Standards Organization/Open Systems Interconnect (ISO/OSI) X.25, X.29, X.28, X.3 protocols;

• Ethernet, Token Ring, High Performance Parallel Interface (HIPPI), and Fiber Distributed Data Interface (FDDI) transports and;

• Industry standard buses, the first of which is VME, to facilitate the integration of third-party communication products."

[TCP/IP, NFS, SNA-3270, 3770, LU6.2/PU2.1, International Standards Organization/Open Systems Interconnect (ISO/OSI) X.25, X.29, X.28, X.3 protocols]
"This is a powerful advantage to KSR if all true." -DW

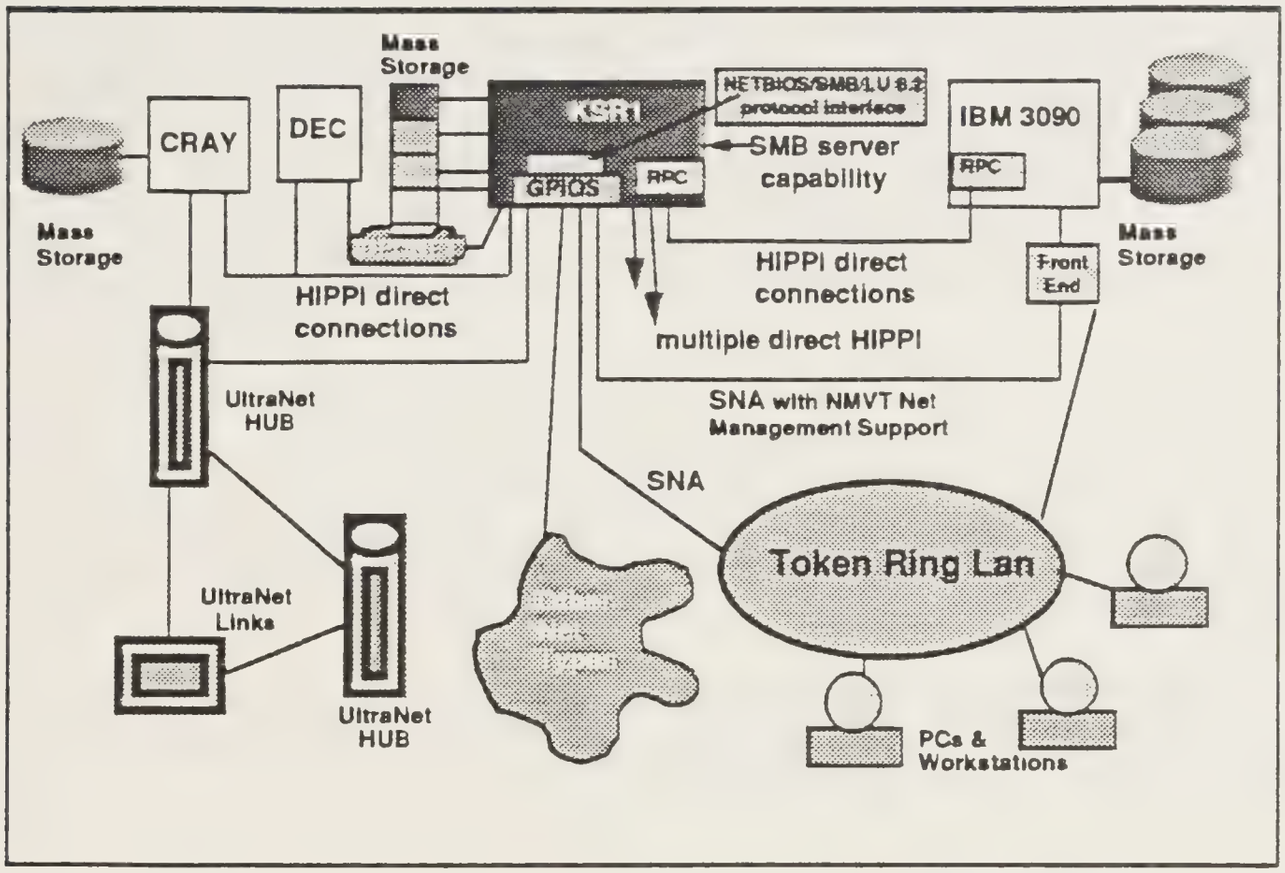
Figure 5 Parallel Program Execution



"As clear as mud." -DW

"Note use of both shared and local/private memory spaces to help keep their locality high and limit the need to synchronize their caches as often." -DD

Figure 6 Co-existence



KSR1 Mass Storage

"The KSR1 mass storage subsystem is a direct extension of the ALLCACHE memory architecture -making the KSR1 the first highly parallel computer that allows users to view mass storage as a direct extension of memory. ALLCACHE supports single level addressing across all KSR1 mass storage subsystems. The memory system is responsible for moving data between mass storage files and memory. A program can access a mass storage file by memory access "load " and "store" instructions.

"This must make it hard to optimize I/O! Unless they just mean virtual memory."
-Ephraim

"This memory system frees the programmer from creating explicit read and write operations to the mass storage file. The programmer simply views these files as part of a program's address space. At the same time, the programmer is not required to change other programs compatible with KSR1 that use explicit "read " and "write" conventions."

"The KSR1 mass storage-memory interconnection feature is especially important for large database and OLTP applications. They can leverage a linear address space that covers the entire disk subsystem and memory system.

"This is actually kind of nice, if it can achieve performance." -Guy

"Disadvantage: secondary storage capacity is limited." -DW

"Bunk. Nobody writes DB or OLTP applications using a uniform address space. Even if it's a good idea, this is entirely non-standard." -Craig

"The KSR1 mass storage subsystem achieves high availability and high performance by leveraging low cost, high density storage. The KSR1 subsystem uses disk drive systems, configured by the KSR OS operating system, as redundant arrays of inexpensive disks ("RAID'). Users can manage storage capabilities, either through mirroring disk support or N+l redundancy support. Under the N+l scheme, each group of four disks has a fifth parity disk responsible for keeping track of all stored information. In case of failure, the data is reconstructed by software from the remaining disks. Redundancy is used to achieve very high reliability. Additionally, KSR1 mass storage subsystems can split individual files across multiple drives, a method called "file striping." Files as large as 8 Terabytes can be distributed across disk drives for maximum performance."

"How many disks?" -Craig

"The KSR1 mass storage subsystem ranges from 10 to more than 15,000 GBytes, and supports an 1/0 bandwidth speed of up to 15,300 MBytes per second.

"It would be nice to know more..." -DW

A Highly Available System

In addition to redundant mass storage, KSR1 maintains highly available features throughout the system, illustrated in Table 2. A key objective underlying the KSR1 is to ensure that it will restart in a matter of seconds, with data integrity guaranteed at all times. Redundant support and on-line replacement are available throughout the system in processor modules, the disk subsystem, the power system and 1/0 subsystem. KSR1 high availability features include:

- Automatic fault detection, isolation and containment to prevent a single failure from causing a complete system failure;

- On-line monitoring, diagnostic and maintenance support;

- Fault tolerant database support;

- Battery back-up power system which will keep an entire system running for up to 5 minutes after a power interruption, at which point the system performs a graceful automatic shutdown;

- Automatic system reconfiguration after a system failure.

[Automatic fault detection, isolation and containment...]

"What if your variable is in a local p.e.'s memory? Do you have to mirror everything if you want to not lost data?" -DW

Table 2 KSR1 High Availability

Hardware Component	Recovery: Redundant	Recovery: Online Replacement	Recovery: By-pass for continued operation
Batteries	Yes	Yes	Yes
Disk subsystem Including power distribution and fans	Yes	Yes	Yes
I/O subsystem	Yes	Yes	Yes
APRD Cell	Yes	Yes	Yes

Summary

The KSR1 offers a complete and highly parallel computer system for production environments. The KSR1 has, for the first time, eliminated the trade offs associated with today's high performance systems, by coupling the power of highly parallel computing technology with the familiar shared memory programming model traditionally used by mainframes and supercomputers. On this platform, users can work with a broad range of familiar software tools including a fully implemented UNIX operating system compatible with OSF/1, COBOL, Fortran, C languages and ORACLE RDBMS software. Kendall Square Research has achieved this industry milestone through the invention of ALLCACHE, the first memory architecture to deliver the conventional, sequentially consistent shared memory programming model in a highly parallel computer. Because of the company's system design innovations, the KSR1 enables users to economically bring the absolute performance and price/performance advantages of parallel processing to a full range of commercial and technical applications.

Additional responses:

John Richardson:

1) I don't believe the dusty deck hype!

In this sense the CM5 can *also* run dusty decks! ANY application that currently runs on SUN's including Oracle, Lotus123, MSC Nastran,... will also run on the CM5 (without even recompiling!!!). Of course it will run on the partition manager, but it *will* run.

KSR can get to this stage only after they first port the application to their proprietary processor (in some cases a *highly* non-trivial task). These old codes were written for a single thread of control and *will not* be able to use more than one processor at a time *without* jumping through all the hoops we've been jumping through since the CM-1.

2) There seems to be a lack of discussion about timesharing groups of processors. If all of the processors are running OSF how does it work? Users experience with CRAY YMP-8's (a shared memory machine) has shown us that you always end up using one processor *unless* you have the entire system to yourself.

3) I did find the APRD cell and its Linpack,Livermore Loops, and Matmul numbers impressive.

In summary I believe this technology is a lot of hype designed in a vacuum. There is no free lunch.

Steve Smith:

KSR ACHIEVES 43GFLOPS AND RUNS DUSTY DECKS
They imply that they can run dusty decks at fast speeds but this is not substantiated. They may have a very fast machine and they may be able to run dusty decks but they may not be able to run dusty decks fast.

MAKING MEMORY APPEAR TO BE A UNIFORM ADDRESS SPACE IS ALL THAT IS IMPORTANT FOR A SHARED ADDRESS MACHINE
The usual big problems with a shared memory occur when two users want the same data at the same time - or are both modifying it at the same time. They have one line about their locking and unlocking sheme and mention how data is moved "close" to the processor using it. I think that in our experience this is not where the problem occurs.

ALLCACHE IS A GOOD IDEA JUST AS WAS VIRTUAL MEMORY
Perhaps, but implementing virtual memory caused little or no performance

degredation for users. Hiding the concepts of "near" and "far" memory accesses may result in some simplification for the programmer but also may dramatically decrease performance with no easy way to get it back. KSR has not made all memory accesses equally fast they have just blinded the user to the fact that some are slow and some are fast.

SCALABILITY
They have an appealing story of using a simple ring communications path but in reality they have a hierarchy and it is not clear what higher levels above the current 2 would look like or how they would perform.

PARALLEL SYSTEMS TAKE MORE EFFORT TO PROGRAM
Point out some of our examples like the Census project where programming in parallel is more natural and more efficient than programming in serial.

KSR IS A MAINSTREAM COMPUTER AS EVIDENCED BY THEIR THIRD PARTY INVOLVEMENT
This is their most compelling argument. Our best reply is that we have actually sold machines to industry not just for research - Amex, DJ, Mobil.

Denny Dahl:

I received your cover letter and print-out of the KSR technical description. I read the description, and made a comment on it, and let the whole thing simmer in my mind for a while. The conclusion I came to is that to try to rebut their claims on a point by point basis is very hard. This is because a lot of their claims are not quantifiable, or verifiable in any real sense. And lots of their claims are just wrong.

For example, on page 1, they say :

> "The ALLCACHE is the first memory architecture to deliver the conventional, sequentially consistent shared memory programming model in a highly parallel computer."

This is wrong. Fortran-90 is a language built on exactly the same model : a conventional, sequentially consistent shared memory programming model. And, obviously, we deliver this programming model to *our* users.

This point (that KSR is the *first* to offer the shared memory model) is made much of throughout the first few pages. I count two references to it on page 3, and a blatant lie on page 6 :

> "... because their (that is, TMC's and Intel's memory architectures cannot support conventional shared memory

programming."

Generously, we could respond by gently correctly the error that KSR makes in these statements, and by welcoming them to the ever-growing family of companies that are providing shared memory programming models on distributed memory machines. One of the first companies at the head of this list is CRAY.

At this point, you must think I've slipped a ratchet. But, I believe it is important to stick to technical realities in this discussion, since that is where we are on the firmest ground. And the fact of the matter is that *all* vector supercomputers use a physically distributed memory. That's because the processor speed in such a machine is so much greater than the memory speed that it is *always* necessary to couple together a number of memory units to provide the necessary memory bandwidth to keep the processors fed with data.

A CRAY, depending on memory configuration, can have up to 64 banks of memory. I believe the number in the C90 will be more like 256. The point to make here is that the hardware notion of synthesizing a single logical memory out of a large number of physically distinct units is a very old idea.

The difference in the way that *we* use distributed memory and the way CRAY uses it is that at the OS level, we don't have a single address space, and CRAY does. But at the language level, there is *no* difference. Both we and CRAY support shared memory models. We happen to do it natively in F90 and they do it in F77, but this is merely a language choice issue and has *nothing* to do with programming models.

In fact, at this level of discussion, one can make a clean distinction between us and Intel. You might counter that Intel also fits into the category of distributed memory architectures. This is certainly true, and I would agree with this statement. But the big difference between us and Intel (from the programming model point of view) is that Intel does *not* have a shared memory programming model (ie. no Data Parallel) and we *do*.

The above point is important, and worth making loudly and clearly. However, there is another point which is equally important, and we should not allow it to be lost in the debate. And one should be careful with this point since there is definite technical content to it. This point revolves around scalability, and in particular,

scalability as it applies to their "network".

The KSR publicity invites the reader/listener to believe that ``magic'' is possible. This is just wrong. There is *no* magic. What I mean is that KSR would like the user to believe that through the ``magic'' of ALLCACHE, all the issues associated with building a distributed memory machine just ``go away'', and everything ``just works''. In fact, the KSR architecture cannot escape from the fundamental limitations of building a distributed memory machine, any more than it can somehow evade the laws of physics.

Communication amongst the distributed memories of *any* distributed memory machine is a fact of life. And the quality of life is then determined by how well the hardware supports the communication. The fundamental fact is that ring-based hardware is *not* scalable. Scalability means (among other things)that the bisection bandwidth of your machine should be proportional to the number of processors in the machine. This requirement is simply not met by a ring network. The same statement applies to grids, and for this reason I would also say that the Paragon (Intel) is not a scalable architecture.

(Technical aside : the bisection bandwidth of a ring is O(1) and the bisection bandwidth of a grid is O(N^(1/2)). The goal is to have bisection bandwidth which is O(N), which *we* achieve.)

No matter how dynamic the mapping between addresses and cache locations, the fact is that all real applications will cause communication. It doesn't matter if you call the communication GATHER/SCATTER, or SEND/GET or ALLCACHE. And the machine that does the best at supporting this communication from a hardware level through a language level will always win. In the absence of any practical demonstrations of the power of the ring network, and in light of the well-known theoretical results on the limitations of ring topologies, I would recommend that customers take a ``show-me'' attitude about the real performance that KSR can bring to the table.

I want to see performance numbers for an unstructured finite element code running on the KSR machine before I believe *any* of their claims.

As an addendum, I'd like to advise caution on our criticism of the good idea that KSR has brought to the table. This is the notion that one might want a dynamic translation function between addresses and processors/memories. Note that this

idea is *not* new. In fact, if you look at Danny's book, ``The Connection Machine'', section 6.8, this precise idea is described and called ``swapping''. I wouldn't like to criticize this idea, since we may end up adopting some variant of it ourselves someday.